# EAST Search History

| Ref # | Hits | Search Query | DBs | Default Operator | Plurals | Time Stamp |
|---|---|---|---|---|---|---|
| S1 | 4 | eduard near de near jong.in. | US-PGPUB; USPAT | OR | ON | 2007/01/23 08:46 |
| S2 | 85 | eduard near "de jong".in. | US-PGPUB; USPAT | OR | ON | 2007/01/23 08:47 |
| S3 | 7054 | "sun microsystems".as. | US-PGPUB; USPAT | OR | ON | 2007/01/23 08:46 |
| S4 | 2 | S3 and (obfuscat$4 and opcode).clm. | US-PGPUB; USPAT | OR | ON | 2007/01/23 08:50 |
| S5 | 5 | S3 and (obfuscat$4 and counter).clm. | US-PGPUB; USPAT | OR | ON | 2007/01/23 08:50 |
| S6 | 28 | ("5057997" "5136705" "5367687" "5463746" "5524256" "5636352" "5659754" "5819117" "5828853" "5905876" "5968164" "5999732" "6081665" "5887161").pn. | US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB | OR | ON | 2007/01/23 08:51 |
| S7 | 297 | 713/190.ccls. | US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB | OR | ON | 2007/01/23 09:41 |
| S8 | 35 | S7 and obfuscat$4 | US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB | OR | ON | 2007/01/23 09:56 |
| S9 | 1505 | 717/106-109.ccls. | US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB | OR | ON | 2007/01/23 09:57 |
| S10 | 43 | S9 and opcode | US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB | OR | ON | 2007/01/23 10:10 |
| S11 | 3 | S9 and (instruction near counter) | US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB | OR | ON | 2007/01/23 10:15 |

# EAST Search History

| | | | | | | |
|---|---|---|---|---|---|---|
| S12 | 287 | obfuscat$4 and opcode | US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB | OR | ON | 2007/01/23 10:16 |
| S13 | 155 | S12 and (instruction near3 counter) | US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB | OR | ON | 2007/01/23 10:16 |
| S14 | 147 | S13 and (@pd<"20030925" or @ad<"20030925" or @prad<"20030925" or @rlad<"20030925") | US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB | OR | ON | 2007/01/23 14:26 |
| S15 | 30 | ("4278837" \| "4446519" \| "4465901" \| "4562306" \| "4644493" \| "4683968" \| "4685055" \| "4757534" \| "4817140" \| "4959861" \| "5109413" \| "5123045" \| "5148534" \| "5222134" \| "5287408" \| "5327563" \| "5365586" \| "5390297" \| "5416840" \| "5666411" \| "5671275" \| "5671412" \| "5675645" \| "5825890" \| "6088452" \| "6101606" \| "6134659" \| "6266416" \| "6334189" \| "6480959").PN. | US-PGPUB; USPAT; USOCR | OR | ON | 2007/01/23 10:42 |
| S16 | 95 | ("20050069131" "20050069138" "20050071652" "20050071653" "20050071655" "20050071664" "4961191" "5367149" "5802071" "5615332" "4767972" "4777589" "5016212" "5019970" "5381534" "5404539" "5544359" "5561763" "5675773" "5701494" "5784276" "5794240" "5822583" "5945986" "5970250" "5995987" "6038396" "6075529" "6107874" "6157997" "6212497" "6212513" "6212497" "6212513" "6820051" "20020008717" "20030061274" "20040255124" "20050132064" "20050257054" "5237668" "5481693" "5781750" "5287490" "5428786" "5742804" "4949250" "5274826" "5317740" "5388233" ).pn. | US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB | OR | ON | 2007/01/23 12:42 |

# EAST Search History

| | | | | | | |
|---|---|---|---|---|---|---|
| S17 | 21 | (US-20050071655-$).did. or (US-5463746-$ or US-6842862-$ or US-6779114-$ or US-6757831-$ or US-6675298-$ or US-6665796-$ or US-6643775-$ or US-6609201-$ or US-6598166-$ or US-6594761-$ or US-6308256-$ or US-6175925-$ or US-6006328-$ or US-7111278-$ or US-6675376-$ or US-5913064-$ or US-6266416-$ or US-6101606-$ or US-6480959-$ or US-6334189-$). did. | US-PGPUB; USPAT | OR | ON | 2007/01/23 14:10 |
| S18 | 10 | S17 and (modulo or mod) | US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB | OR | ON | 2007/01/23 14:22 |
| S20 | 6256 | (modulo or mod) near "n" | US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB | OR | ON | 2007/01/23 14:25 |
| S21 | 70 | S20 with instruction | US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB | OR | ON | 2007/01/23 14:25 |
| S22 | 57 | S21 and (@pd<"20030925" or @ad<"20030925" or @prad<"20030925" or @rlad<"20030925") | US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB | OR | ON | 2007/01/23 14:26 |
| S23 | 21 | ("6289455"\|"6374402"\|"6594761"\|"6615350"\|"6640305"\|"6668325"\|"6779114"\|"6862683"\|"7089594"\|"7127712"\|"7150003").PN. | US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB | OR | ON | 2007/01/23 15:22 |
| S25 | 2 | S17 and (filter$4 near3 instruction) | US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB | OR | ON | 2007/01/23 15:48 |

# EAST Search History

| | | | | | | |
|---|---|---|---|---|---|---|
| S26 | 2 | "5564111".pn. | US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB | OR | ON | 2007/01/24 15:44 |

Terms used **executing obfuscated program** Found **53,748** of **195,947**

Sort results by: relevance
Display results: expanded form
Save results to a Binder
Search Tips
Open results in a new window
Try an Advanced Search
Try this search in The ACM Guide

Results 21 - 40 of 200 Result page: previous 1 **2** 3 4 5 6 7 8 9 10 next
Best 200 shown Relevance scale

**21** Modeling methodology B: modeling and simulation of computer systems: Performance analysis of binary code protection
David M. Nicol, Hamed Okhravi
December 2005 **Proceedings of the 37th conference on Winter simulation WSC '05**
**Publisher:** Winter Simulation Conference
Full text available: pdf(159.34 KB) Additional Information: full citation, abstract, references

Software protection technology seeks to prevent unauthorized observation or use of applications. Cryptography can be used to provide such protection, but imposes a potentially significant additional computation load. This paper examines the performance impact of two software protection techniques. We develop an analytic model and validate it using a detailed discrete-event simulator applied to memory reference traces of well-known benchmark programs. We find that even though the added workload m ...

**22** A tentative approach to constructing tamper-resistant software
Masahiro Mambo, Takanori Murayama, Eiji Okamoto
January 1998 **Proceedings of the 1997 workshop on New security paradigms NSPW '97**
**Publisher:** ACM Press
Full text available: pdf(1.05 MB) Additional Information: full citation, references, index terms

**23** Session 2: Review and analysis of synthetic diversity for breaking monocultures
James E. Just, Mark Cornwell
October 2004 **Proceedings of the 2004 ACM workshop on Rapid malcode WORM '04**
**Publisher:** ACM Press
Full text available: pdf(356.14 KB) Additional Information: full citation, abstract, references, citings, index terms

The increasing monoculture in operating systems and key applications and the enormous expense of N-version programming for custom applications mean that lack of diversity is a fundamental barrier to achieving survivability even for high value systems that can afford hot spares. This monoculture makes flash worms possible. Our analysis of vulnerabilities and exploits identifies key assumptions required to develop successful attacks. We review the literature on synthetic diversity techniques, f ...

**Keywords**: diversity, n-version programming, vulnerability

**24** Language-based security: SELF: a transparent security extension for ELF binaries
Daniel C. DuVarney, V. N. Venkatakrishnan, Sandeep Bhatkar

August 2003 **Proceedings of the 2003 workshop on New security paradigms NSPW '03**
**Publisher:** ACM Press
Full text available: pdf(1.05 MB) Additional Information: full citation, abstract, references

The ability to analyze and modify binaries is often very useful from a security viewpoint. Security operations one would like to perform on binaries include the ability to extract models of program behavior and insert inline reference monitors. Unfortunately, the existing manner in which binary code is packaged prevents even the simplest of analyses, such as distinguishing code from data, from succeeding 100 percent of the time. In this paper, we propose SELF, a security-enhanced ELF (Executable ...

**25** Software and languages: Proteus: virtualization for diversified tamper-resistance
Bertrand Anckaert, Mariusz Jakubowski, Ramarathnam Venkatesan
October 2006 **Proceedings of the ACM workshop on Digital rights management DRM '06**
**Publisher:** ACM Press
Full text available: pdf(299.79 KB) Additional Information: full citation, abstract, references, index terms

Despite huge efforts by software providers, software protection mechanisms are still broken on a regular basis. Due to the current distribution model, an attack against one copy of the software can be reused against any copy of the software. Diversity is an important tool to overcome this problem. It allows for renewable defenses in space, by giving every user a different copy, and renewable defenses in time when combined with tailored updates. This paper studies the possibilities and limitation ...

**Keywords**: copyright protection, diversity, intellectual property, obfuscation, tamper-resistance, virtualization

**26** HOIST: a system for automatically deriving static analyzers for embedded systems
John Regehr, Alastair Reid
October 2004 **ACM SIGOPS Operating Systems Review , ACM SIGARCH Computer Architecture News , ACM SIGPLAN Notices , Proceedings of the 11th international conference on Architectural support for programming languages and operating systems ASPLOS-XI**, Volume 38 , 32 , 39 Issue 5 , 5 , 11
**Publisher:** ACM Press
Full text available: pdf(145.53 KB) Additional Information: full citation, abstract, references, citings, index terms

Embedded software must meet conflicting requirements such as be-ing highly reliable, running on resource-constrained platforms, and being developed rapidly. Static program analysis can help meet all of these goals. People developing analyzers for embedded object code face a difficult problem: writing an abstract version of each instruction in the target architecture(s). This is currently done by hand, resulting in abstract operations that are both buggy and im-precise. We have developed Hoist: a ...

**Keywords**: abstract interpretation, object code, program verification, static analysis

**27** Session 11A: On obfuscating point functions
Hoeteck Wee
May 2005 **Proceedings of the thirty-seventh annual ACM symposium on Theory of computing STOC '05**
**Publisher:** ACM Press
Full text available: pdf(333.82 KB) Additional Information: full citation, abstract, references, citings, index terms

We investigate the possibility of obfuscating point functions in the framework of Barak et al. from Crypto '01. A point function is a Boolean function that assumes the value 1 at exactly one point. Our main results are as follows:We provide a simple construction of efficient obfuscators for point functions for a slightly relaxed notion of obfuscation, for

which obfuscating general circuits is nonetheless impossible. Our construction relies on the existence of a very strong one-way permutation, a ...

**Keywords**: obfuscation

**28** Code optimization II: Hiding program slices for software security

Xiangyu Zhang, Rajiv Gupta

March 2003 **Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization CGO '03**

**Publisher:** IEEE Computer Society

Full text available: pdf(1.05 MB) Additional Information: full citation, abstract, references, citings, index terms

Given the high cost of producing software, development of technology for prevention of software piracy is important for the software industry. In this paper we present a novel approach for preventing the creation of unauthorized copies of software. Our approach splits software modules into *open* and *hidden* components. The open components are installed (executed) on an unsecure machine while the hidden components are installed (executed) on a secure machine. We assume that while open ...

**29** Tamper-proofing software watermarks

Clark Thomborson, Jasvir Nagra, Ram Somaraju, Charles He

January 2004 **Proceedings of the second workshop on Australasian information security, Data Mining and Web Intelligence, and Software Internationalisation - Volume 32 ACSW Frontiers '04**

**Publisher:** Australian Computer Society, Inc.

Full text available: pdf(249.93 KB) Additional Information: full citation, abstract, references, citings, index terms

We introduce a novel method called *constant encoding*, which can be used to tamper-proof a software watermark that is embedded in the dynamic data structures of a program. Our novel tamper-proofing method is based on transforming numeric or non-numeric constant values in the text of the watermarked program into function calls whose value depends on the watermark data structure. Under reasonable assumptions about the knowledge and resources of an attacker, we argue that no attacker can be c ...

**30** WBIA'05: Practical analysis of stripped binary code

Laune C. Harris, Barton P. Miller

December 2005 **ACM SIGARCH Computer Architecture News**, Volume 33 Issue 5

**Publisher:** ACM Press

Full text available: pdf(224.79 KB) Additional Information: full citation, abstract, references, index terms

Executable binary code is the authoritative source of information about program content and behavior. The compile, link, and optimize steps can cause a program's detailed execution behavior to differ substantially from its source code. Binary code analysis is used to provide information about a program's content and structure, and is therefore a foundation of many applications, including binary modification[3,12,22,31], binary translation[5,29], binary matching[30], performance profiling[13,16,1 ...

**31** Computer security and encryption II: Some new approaches for preventing software tampering

Bin Fu, Golden Richard, Yixin Chen

March 2006 **Proceedings of the 44th annual southeast regional conference ACM-SE 44**

**Publisher:** ACM Press

Full text available: pdf(124.74 KB) Additional Information: full citation, abstract, references

In this paper, we propose several methods to increase the difficulty of reverse engineering applications, with special emphasis on preventing the circumvention of copy protection mechanisms that permit only authorized users to execute the applications. We

apply the hashing function to transform some constants in the software and recover them during the execution with the correct input of the password. The security of such a method depends on the hardness of the invertibility of the hashing funct ...

32 Trust (and mistrust) in secure applications

John Viega, Tadayoshi Kohno, Bruce Potter
February 2001 **Communications of the ACM**, Volume 44 Issue 2
**Publisher:** ACM Press
Full text available: pdf(91.25 KB) html(33.48 KB) Additional Information: full citation, references, citings, index terms, review

33 Network, online game: A proposal of encoded computations for distributed massively multiplayer online services

Keiichi Endo, Minoru Kawahara, Yutaka Takahashi
June 2006 **Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology ACE '06**
**Publisher:** ACM Press
Full text available: pdf(196.08 KB) Additional Information: full citation, abstract, references, index terms

This paper deals with Massively Multiplayer Online services, which are used to interact with other user in real time. Although it is possible to distribute the load on the server to users' computers, the distributed services are more vulnerable to cheating such as data theft. We propose a method for concealing data from a user who is authorized to manage the data. The method makes it possible to carry out various kinds of operations with data and program codes concealed. We show that concealed i ...

**Keywords**: distributed massively multiplayer online services, encoded computations, interactive real-time applications, peer-to-peer

34 Software Engineering for Secure Systems (SESS) --- Building Trustworthy Applications: Enabling control over adaptive program transformation for dynamically evolving mobile software validation

Mike Jochen, Anteneh Addis Anteneh, Lori L. Pollock, Lisa M. Marvel
May 2005 **ACM SIGSOFT Software Engineering Notes , Proceedings of the 2005 workshop on Software engineering for secure systems—building trustworthy applications SESS '05**, Volume 30 Issue 4
**Publisher:** ACM Press
Full text available: pdf(134.11 KB) Additional Information: full citation, abstract, references, index terms

Many researchers are investigating the use of adaptive program transformation as a way to efficiently improve program performance. Performance improving transformations are performed at runtime to adapt to the possibly changing runtime characteristics of the program. Leveraging this kind of program transformation on multiple hosts can achieve these same performance gains while reducing the overhead to apply the transformations on the local machine running the program. The reduction in overhead i ...

**Keywords**: computer security, dynamic and adaptive program transformation, integrity, mobile code, program analysis

35 Information protection methods: Display-only file server: a solution against information theft due to insider attack

Yang Yu, Tzi-cker Chiueh
October 2004 **Proceedings of the 4th ACM workshop on Digital rights management DRM '04**
**Publisher:** ACM Press
Full text available: Additional Information:

pdf(311.80 KB) full citation, abstract, references, index terms

Insider attack is one of the most serious cybersecurity threats to corporate America. Among all insider threats, information theft is considered the most damaging in terms of potential financial loss. Moreover, it is also especially difficult to detect and prevent, because in many cases the attacker has the proper authority to access the stolen information. According to the 2003 CSI/FBI Computer Crime and Security Survey, theft of proprietary information was the single largest category of los ...

**Keywords**: access, digital rights management, information theft, insider attack

36 Defensive technology: Detection of injected, dynamically generated, and obfuscated malicious code

Jesse C. Rabek, Roger I. Khazan, Scott M. Lewandowski, Robert K. Cunningham
October 2003 **Proceedings of the 2003 ACM workshop on Rapid malcode WORM '03**
**Publisher:** ACM Press

Full text available: pdf(240.68 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents DOME, a host-based technique for detecting several general classes of malicious code in software executables. DOME uses static analysis to identify the locations (virtual addresses) of system calls within the software executables, and then monitors the executables at runtime to verify that every observed system call is made from a location identified using static analysis. The power of this technique is that it is simple, practical, applicable to real-world software, and high ...

**Keywords**: anomaly detection, code analysis, dynamic analysis, execution monitoring, intrusion detection, malicious code detection, static analysis, system calls

37 Security and protection: On approximate matching of programs for protecting libre software

Arnoldo José Müller Molina, Takeshi Shinohara
October 2006 **Proceedings of the 2006 conference of the Center for Advanced Studies on Collaborative research CASCON '06**
**Publisher:** ACM Press

Full text available: pdf(295.98 KB) htm(1.60 KB) Additional Information: full citation, abstract, references

Libre software licensing schemes are sometimes abused by companies or individuals. In order to encourage open source development it is necessary to build tools that can help in the rapid identification of open source licensing violations. This paper describes an attempt to build such tool. We introduce a framework for approximate matching of programs, and describe an implementation for Java byte-code programs. First, we statically analyze a program to remove dead code, simplify expressions and t ...

38 String analysis for x86 binaries

Mihai Christodorescu, Nicholas Kidd, Wen-Han Goh
September 2005 **ACM SIGSOFT Software Engineering Notes , The 6th ACM SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering PASTE '05**, Volume 31 Issue 1
**Publisher:** ACM Press

Full text available: pdf(188.04 KB) Additional Information: full citation, abstract, references, index terms

Information about string values at key points in a program can help program understanding, reverse engineering, and forensics. We present a static-analysis technique for recovering possible string values in an executable program, when no debug information or source code is available. The result of our analysis is a regular language that describes a superset of the string values possible at a given program point. We also impart some of the lessons learned in the process of implementing our analys ...

39 Technical contributions: On readability of programs with loops

Janusz W. Laski
November 1979 **ACM SIGPLAN Notices**, Volume 14 Issue 11
**Publisher:** ACM Press
Full text available: pdf(572.16 KB) Additional Information: full citation, references, citings

40 (How) can mobile agents do secure electronic transactions on untrusted hosts? A survey of the security issues and the current solutions

Joris Claessens, Bart Preneel, Joos Vandewalle
February 2003 **ACM Transactions on Internet Technology (TOIT)**, Volume 3 Issue 1
**Publisher:** ACM Press
Full text available: pdf(197.96 KB) Additional Information: full citation, abstract, references, citings, index terms

This article investigates if and how mobile agents can execute secure electronic transactions on untrusted hosts. An overview of the security issues of mobile agents is first given. The problem of untrusted (i.e., potentially malicious) hosts is one of these issues, and appears to be the most difficult to solve. The current approaches to counter this problem are evaluated, and their relevance for secure electronic transactions is discussed. In particular, a state-of-the-art survey of mobile agen ...

**Keywords**: Mobile agent security, electronic transactions, malicious hosts